An Autonomous Path Planner Implemented on the Rocky7 Prototype Microrover

Sharon L. Laubach
Jet Propulsion Laboratory
California Institute of Technology
198-219, 4800 Oak Grove Drive
Pasadena, CA 91109
Sharon.Laubach@jpl.nasa.gov

Joel Burdick
Dept. of Mechanical Engineering
California Institute of Technology
Mail Code 104-44
Pasadena, CA 91125
jwb@robby.caltech.edu

Larry Matthies
Jet Propulsion Laboratory
California Institute of Technology
107, 4800 Oak Grove Drive
Pasadena, CA 91109
Larry.Matthies@jpl.nasa.gov

Abstract:

Much prior work in mobile robot path planning has been based on assumptions that are not really applicable for exploration of planetary terrains. Based on the first author's experience with the recent Mars Pathfinder mission, this paper reviews the issues that are critical for successful autonomous navigation of planetary rovers. No currently proposed methodology accurately addresses all of these issues. We next report on an extension of the recently proposed "Tangent Bug" algorithm. The implementation of this extended algorithm on the Rocky 7 Mars Rover prototype at the Jet Propulsion Laboratory is described, and experimental results are presented. In addition, experience with the limitations encountered by the Sojourner rover in actual Martian terrain suggest that terrain traversability must be more accurately handled in autonomous planning algorithms for interplanetary rovers.

1. Introduction

Planetary scientists have a unique problem when collecting data for their subject: in the absence of an interplanetary manned space program, they must rely upon remote sensing techniques. An invaluable part of their arsenal is spacecraft which can study the region of interest in detail. Landers in particular are able to bring sensors and instruments into close contact with the planetary environment, and to take direct measurements of properties such as soil mechanics and elemental composition as well as being able to image surface features in greater detail than is possible with orbiters or Earth-bound instruments. Landers, however, have the shortcoming that they are limited to a single site for study. An important addition, then, is a mobile robot which can rove over a much larger segment of the terrain and can carry imagers and other instruments to a variety of features spread over this larger area. The recent Mars Pathfinder mission demonstrated the utility of such an approach: the Sojourner rover carried by the lander to the surface of Mars ranged over an area approximately 20 meters square. The Sojourner rover conducted soil experiments in several different terrains, took detailed images of rocks and soils from centimenters away, and placed its on-board spectrometer on 16 distinct targets (9 rocks and 7 soil sites) [7]. Future missions plan to expand this successful technology by incorporating mobile robots ("rovers") which are capable of traversing even larger distances, carrying their instruments to a wider variety of features and even caching samples along the way. A key requirement for these new planetary robots is greater navigational autonomy since longer distances must be covered between opportunities to communicate with Earth than in prior missions.

1.1 The "rover problem"

The "rover problem" is this: Given an unknown, rough terrain, the mobile robot (or "rover") must autonomously navigate from its current position to the goal (e.g. a specified location). The rover uses its sensors to determine the nature of its local environment, and plans its path accordingly. Rather than addressing all of the issues which percolate out of this complex problem, this paper focusses on the aspects relevant to autonomous path planning.

1.2 Relevant work

Much of the prior work for mobile robot path planning has assumed indoor environments and omnidirectional motion and sensing for the robot. The classical path planners described in Latombe [5] further assume a completely known environment. However, the classical planners have the useful properties of correctness and completeness. That is, the paths produced by these planners lie completely in the free space of the robot, and the planners will generate a path if one is possible, or will halt otherwise. The class of heuristic planners, such as Brooks' subsumption architecture [1] or the track

arbitration schemes developed at CMU [4, 9], as well as the "Go To Waypoint" algorithm employed by the Sojourner rover [7], share the useful property of being able to be made sensor-based much more easily than the classical planners, and can be applied to unknown terrains. However, although they are designed to work well in "most" environment configurations, they lack completeness--there is no guarantee that the algorithm will halt, or that the robot will be able to find the goal even if a path exists. In addition, the heuristic planners tend to result in lengthy paths in natural terrain (relative to the optimal path), as has been seen in field trials with the Rocky7 rover using a similar "Go To Waypoint" algorithm. (Most traverses of the Sojourner rover on Mars have not been long enough to demonstrate this effect.) It is desirable, then, to develop a path planner which combines the best of both worlds: good local properties with a global convergence criterion. Such planners fall under the aegis of what can be called "complete and correct" sensor-based motion planning. These algorithms are incremental in nature: the robot senses its immediate environment, then determines the "best" local path segment based upon these measurements. After moving along the local path, the robot begins the cycle again with its sensors.

Using this model, two distinct approaches have been explored, both of which adapt classical methods to a local "visible area": methods which build "roadmaps" within the free space in the visible area, such as Choset's HGVG [2] and the "TangentBug" algorithm of Kamon, et al. [3]; and those which utilise approximate cell decomposition, filling in a grid-based world model incrementally, such as Stentz' D* algorithm. [9,10] Both D* and TangentBug produce "locally optimal" paths (i.e., the paths are the shortest possible using solely local information). D* has the advantage of being well-developed and of being suited well to rough terrain navigation. However, the grid-based world model requires a significant amount of memory for storage. In particular, the completeness of D* depends entirely upon accuracy in its world model, which is determined by cell granularity. In any case, it would be useful to develop an alternate planner which produces locally optimal paths through rough terrain without requiring an extensive world model.

The authors' experience with path planning on the Sojourner rover (and the similar planner on Rocky7) indicated that a substantially improved algorithm would be necessary to fulfill the objectives of future missions. The approach taken in this paper is to extend the "TangentBug" algorithm developed by Kamon, Rivlin, and Rimon [3] to be able to account for characteristics of a planetary rover operating in rough terrain. The next section will provide a brief review of the original TangentBug algorithm, followed by a discussion of the issues inherent in adapting the algorithm to a "real world" environment. Section 3 describes in detail the specific implementation of an extended version of TangentBug on the Rocky7 prototype microrover developed at the Jet Propulsion Laboratory (JPL), and section 4 presents experimental results from the Rocky7 implementation. Section 5 discusses a proposed extension of the current implementation to deal with the important concept of traversability, which breaks away from the current binary obstacle model (i.e. a point in the enviroment is either an obstacle or lies in free space). More generalised obstacle descriptions are an important research issue in their own right, particularly as path planners are extended to natural terrains. Finally, section 6 contains concluding remarks.

2. TangentBug

The TangentBug algorithm [3] arose out of an attempt to "sensorise" the classical planner based upon the tangent graph [4]. The classical version assumed complete knowledge of the environment, and guaranteed that it would produce the shortest possible path, if a path existed. TangentBug extends the idea of utilising the tangent graph by defining "local tangent graphs" which are entirely contained within the area visible to the robot's sensors. These local graphs are then searched for the locally shortest path. In order to maintain the original planner's desirable property of completeness, a global criterion is added in order to guarantee convergence to the goal. This criterion, direct distance from the goal, and how it is applied within the algorithm, is discussed in the sequel.

2.1 The Original TangentBug Algorithm

The TangentBug algorithm as originally developed by Kamon, Rivlin, and Rimon [3] is designed for a point robot, with omnidirectonal motion and sensing capabilities, in a two-dimensional indoor environment (i.e. flat floors, with obstacles which block both motion and sensing). The limited sensor range of the robot is modelled by a "visibility polygon" (Fig. 1), a star-shaped set which encompasses all points in free space which lie within a radius R of the robot and which can be connected to the robot by a ray lying entirely within the closure of free space. The local tangent graph (LTG) is constructed as follows: the nodes of the graph consist of the current robot position as well as the endpoints of the intersection of the boundaries of sensed obstacles with the circle of radius R around the robot (B(x,R)). An additional node is created at the intersection of the ray between the current robot position and the goal and B(x,R), as long as the

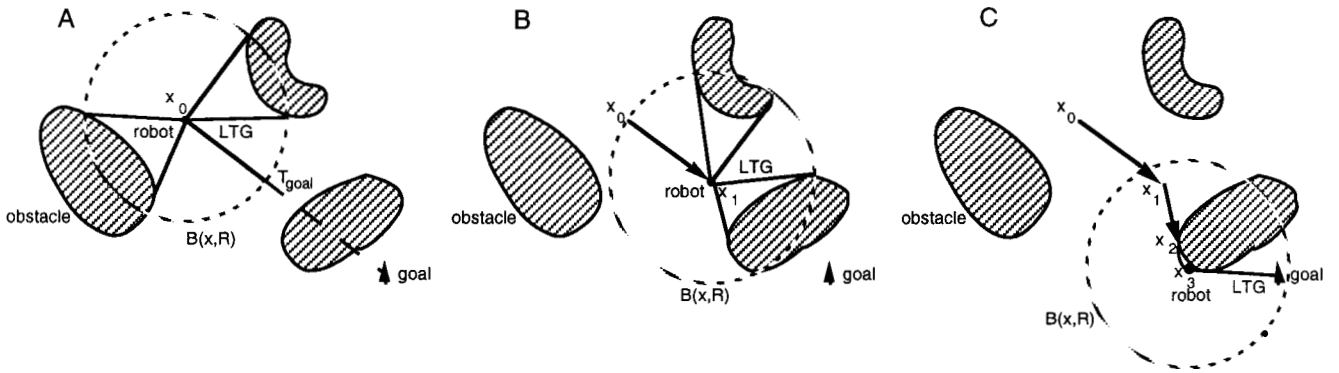

Figure 1. Three stages of the TangentBug algorithm. A) The robot begins at x_0, computes the LTG, then moves along the locally optimal direction, which brings the robot to x_1. B) The LTG at x1. C) The robot has moved to x3 by skirting around the obstacle boundary , and the LTG now contains the goal.

intersection point lies within the closure of the visibility polygon. In effect, this action "projects" the goal position onto the visibility polygon. The edges of the LTG lie wholly within the closure of free space, and are tangent to any obstacles encountered.

The global criterion mentioned earler is the Euclidean distance between the current robot position and the goal, d(x,G), measured without regard to blocking obstacles. The algorithm, then, consists of two behaviours--motion-to-goal, and boundary-following--designed such that the robot is guaranteed to converge to the goal position. The robot begins with motion-to-goal behaviour, in which the LTG is constructed and searched for the optimal local path to the goal position. Outside of the current visibility polygon, the environment is assumed to consist entirely of free space. In fact, the algorithm does not search the entire LTG, but rather a subset, $G1 = \{ V \in LTG \mid d(V,G) \leq \min(d(x,G), d_{Leave}) \}$, where V denotes a node of the LTG and d_{Leave} is a bookmarking parameter initialised at d(start,G). Thus the resultant local path is guaranteed to bring the robot closer to the goal position. After moving a step along the local path, the robot re-senses its environment and begins the cycle again.

The second behaviour, boundary-following, is invoked whenever the robot has arrived at a local minimum, i.e. G1 is empty. The algorithm initialises $d_{min}(x,G)$, then chooses a direction to follow around the obstacle boundary and continues in this direction, using G1 to construct local shortcuts and recording $d_{min}(x,G)$, until either the "leaving condition" is met or the robot detects that it has circumnavigated the obstacle. If a loop is detected, the algorithm halts: the goal is not reachable, and in fact lies within the sensed obstacle. The "leaving condition" is as follows: $\exists V \in LTG$ such that $d(V,G) \leq d_{min}(x,G)$. Once this condition is met, the robot resumes motion-to-goal behaviour. Convergence is thus guaranteed: during motion-to-goal behaviour, d(x,G) decreases monotonically; during boundary-following, the robot leaves the obstacle boundary only when there exists a node in the LTG which lies closer to the goal than any point on the obstacle boundary. In addition, the path length is finite, assuming finite obstacle boundaries, since there are a finite number of both motion-to-goal and boundary-following segments, each of which terminates after finite length.

2.2 Issues in adapting TangentBug to the real world

Though TangentBug incorporates both good local properties--generating locally optimal paths from sensory input--and global convergence to the goal position, several thorny issues arise when the algorithm is applied to planetary rovers. To begin, rovers are not point robots with omnidirectional capabilities. Further, the algorithm assumes that the rover has ideal dead-reckoning ability in order to recognise both the goal position and to determine whether the rover has circumnavigated an obstacle. Yet, as has been demonstrated with the Sojourner rover, it is difficult for a rover to maintain an accurate model of its position within a global coordinate frame without regular updates from operators on Earth. [7] In addition,TangentBug assumes omnidirectional sensing capabilities, which is not always practical for planetary rovers. The Sojourner rover, for example, has a laser striping system which detects obstacles in only the forward direction.

Other difficulties arise from the environment assumed by the TangentBug algorithm. In the typical terrain encountered in the martian environment (Fig. 2), there exist obstacles which block motion, but not sensing (i.e. the rover has the ability to "see over" many obstacles blocking its path)--thus, the visibility polygon is no longer strictly a star-shaped set. In addition, it may be

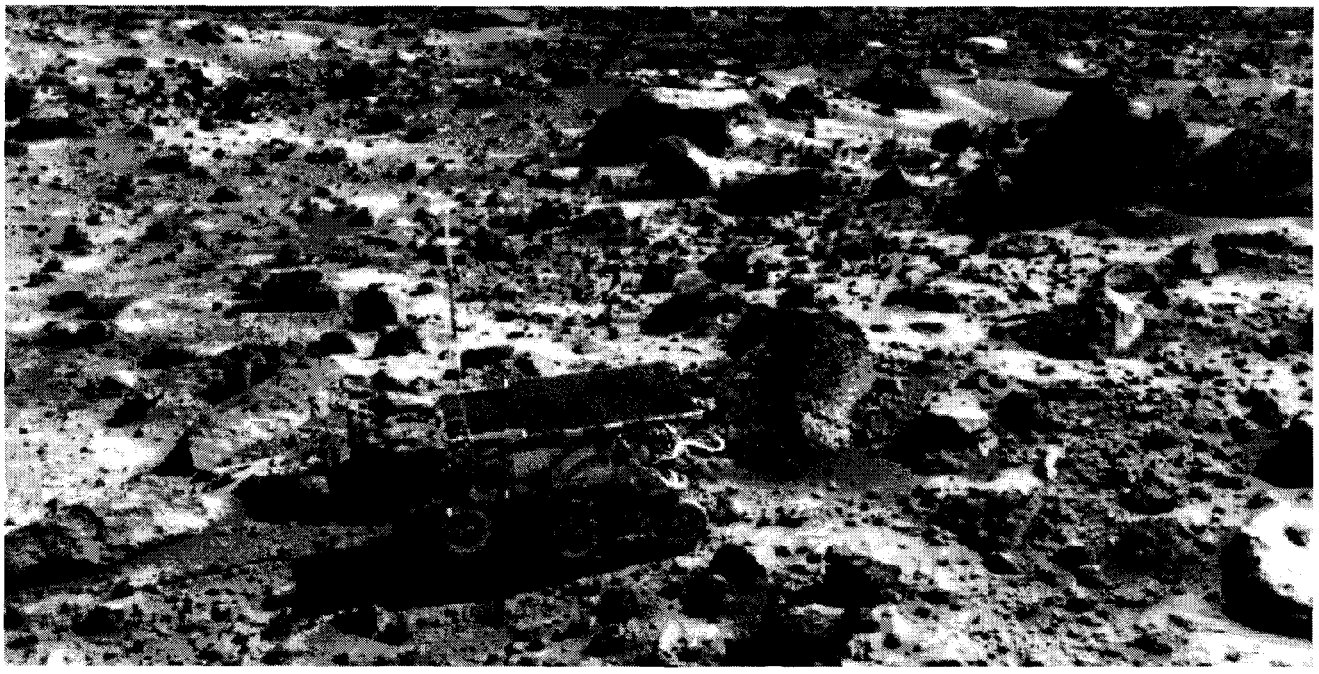

Figure 2. Typical terrain encountered on Mars by the Sojourner rover. The rover is 68 cm long by 48 cm wide, and stands 28 cm tall.

desirable to define the goal in terms other than strictly as a coordinate point. For example, alternative goal designations may include a given directional heading with a distance and/or time limit; a specified terrain feature; or even a motion termination condition derived from an evaluation of data sensed by the rover's science instrument payload during traverse. Further, the current algorithm assumes a binary obstacle model--a point in the environment is either an obstacle or lies in free space. It would be very desirable to be able to incorporate a notion of traversability, in which certain areas could be designated as "soft" obstacles which could be traversed if need be. Thus, the rover could avoid taking an extremely long route--or worse, declaring a goal position unreachable--when it could have chosen to climb over a rough area it would normally have avoided to reach the goal.

Lastly, some of the issues sparked by TangentBug are derived from other considerations. For example, while TangentBug has the attractive feature of producing locally optimal paths, the paths generated by the algorithm skirt the obstacles encountered--a safety hazard for the rover, particularly since its dead-reckoning ability is less than ideal. Also, the boundary-following behaviour is difficult to implement with forward-looking stereo vision sensors as are used on the microrovers developed at JPL. Finally, work must be done to ensure that the details of implementation do not invalidate the conditions of convergence from the original algorithm.

3. Details of specific implementation on Rocky7

The TangentBug algorithm described above has been implemented for the Rocky7 prototype microrover (Fig. 3), which has been developed at the Jet Propulsion Laboratory as a research vehicle to test technologies for future missions [11]. It should be noted that this is the first implementation of TangentBug, or any similar planner, on an actual vehicle. The rover is 60 cm long by 40 cm wide, and stands 33 cm tall, roughly the same size as the Sojourner rover currently on Mars, and carries several science instruments as well as an arm with clamshell scoops for sampling operations. Its mobility chassis consists of a rocker-bogie system able to surmount obstacles 1 1/2 wheel diameters in height (1 wheel diameter = 13 cm). For navigation, it features three pairs of stereo cameras--two mounted below the solar panel (one each front and rear), and one on the deployable 1 m mast--as well as a sun sensor for absolute heading measurements and sensors for measuring odometry for dead reckoning. In addition, although not currently used in this implementation, a localisation algorithm has recently been developed which utilises mast imagery, greatly aiding the rover's dead-reckoning ability.

The path-planner is implemented on this system as several functions which are executed sequentially. The

general scenario is as follows: the rover is situated in rough terrain, with its mast deployed. The operator views the set of panoramic imagery returned by the rover, and designates a goal position. The operator then generates a command sequence which directs the rover to look toward the goal position with its mast cameras, generate a range image from the single-wedge stereo imagery (i.e. only one image pair is utilised, rather than panning the mast cameras to generate a full panorama), and begin the path planner. The planner first detects obstacles within the range image, segments out distinct obstacles, and computes their convex hulls. The second function then "grows" the convex hulls to account for the width of the rover, allowing the TangentBug algorithm to plan as if the rover were a point robot. The third function computes the LTG within the visible region and invokes an A* search to find the shortest local path. Finally, the planner returns a sequence of waypoints designating the generated path, which is followed by the on-board path executor before the operator is queried for the next step. It should be noted here that the path planner can be used with varying levels of autonomy: it could be used either to plan a "strawman" path which could be modified by the user before execution by the rover, or as a fully autonomous single-command system as just described. It should also be noted that the boundary-following behaviour is not yet implemented. Besides technical considerations which complicate implementation of this function--the only sensor on board which can sense obstacles next to the rover is the mast camera pair, which

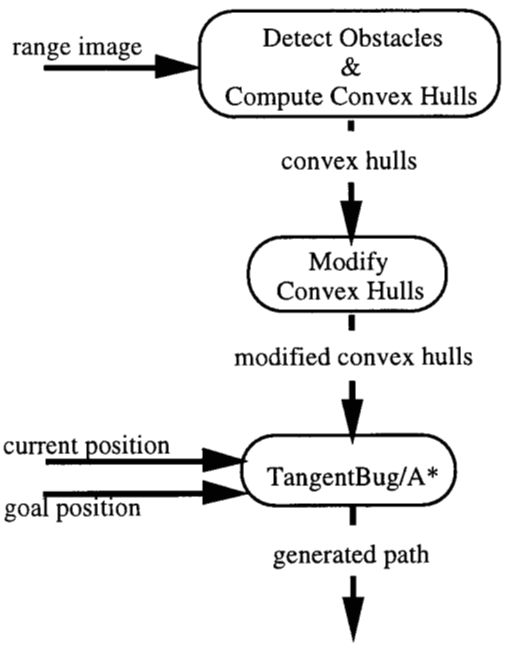

Figure 4. Flow chart of the implementation of TangentBug on Rocky7.

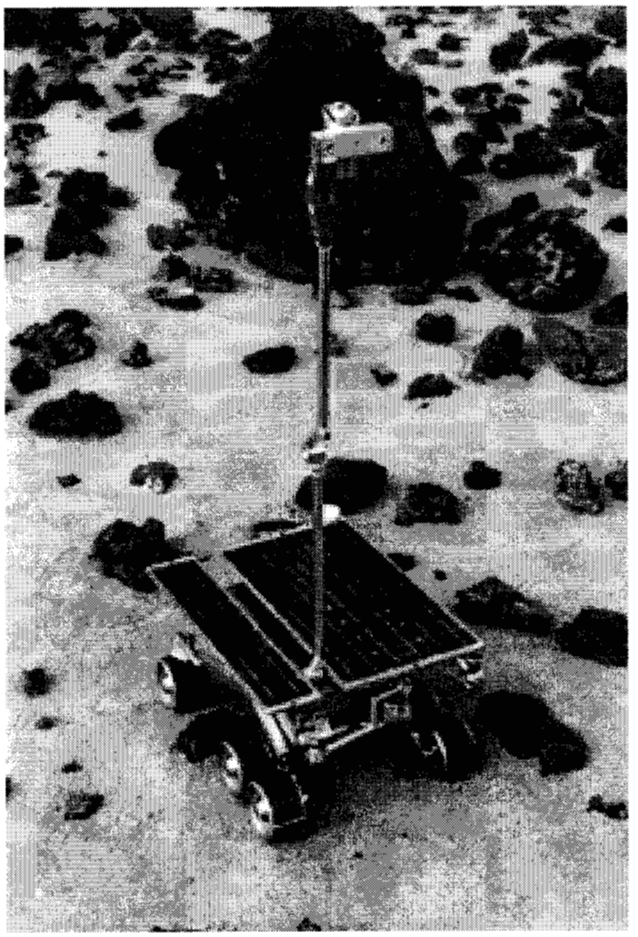

Figure 3. The Rocky7 prototype microrover.

can only be deployed while the rover is stopped--the combination of the expected terrain and the fact that the visible polygon is not strictly a star-shaped set (and thus the LTG is richer) suggests that the boundary following behaviour would not be invoked often.

3.1 Obstacle Detection

The first step of the implementation relies upon the obstacle detection algorithm presented in [5]. This method utilises a simple model of an obstacle--a combination of a step in height and steepness of slope--in order to determine which features in a range image depict obstacles. Additional functions take as input this collection of obstacle points, segment out cohesive regions, and then compute the convex hulls of the distinct regions.

3.2 Modification of Convex Hulls

The convex hulls just computed are then passed to a second function which modifies the obstacle boundaries to

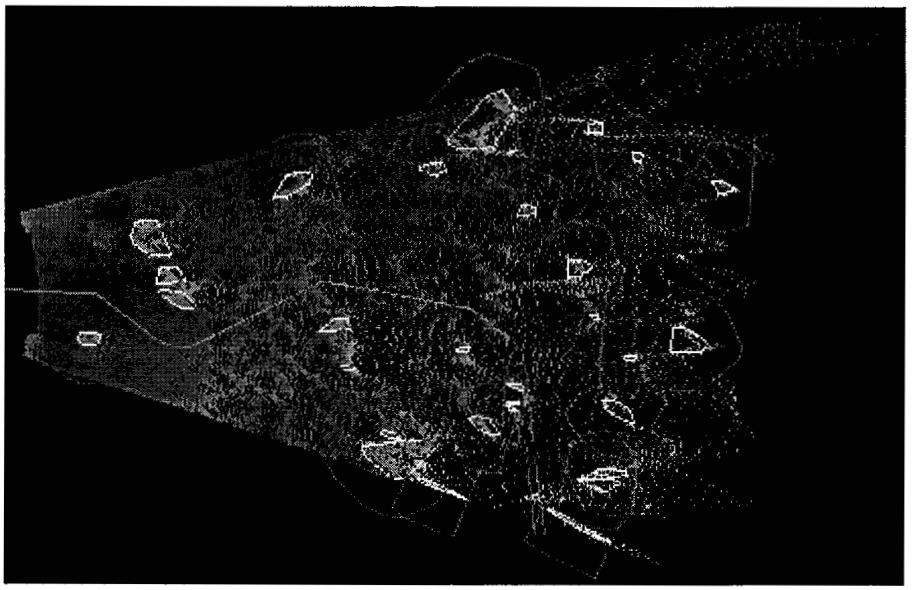

Figure 5. A typical run of the TangentBug implementation using stereo data from the JPL "MarsYard". The plot is an overhead view of the range image generated from stereo imagery; overlays depict salient features. The grey points are pixels which have been marked as obstacles. Each obstacle region is enclosed by a white convex hull; the modified convex hulls are magenta. The goal point has been selected near the center of the panoramic wedge; the autonomously generated path to this point appears in green.

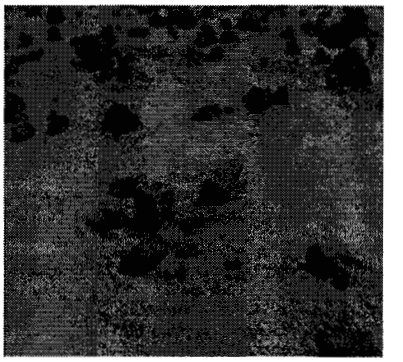

Figure 6. The left image of the stereo pair used to generate the range image at left. The yellow overlay depicts those pixels which have been marked as obstacles.

account for the width of the rover, and to include an empirical safety buffer. This technique is derived from the concept of configuration space, which allows the path planner to assume that the rover is a point robot, thereby easing the computational and conceptual burden. In this particular implementation, each edge of the convex hulls is "pushed outward" by the given buffer width in a direction perpendicular to the orientation of the edge. This action has the effect of doubling the number of vertices in each convex hull, but ensures that the regions on the new obstacle boundary which lie closer to the original convex hull than the buffer width are reasonably small. The choice of buffer width is governed both by the size of the rover, and by considerations of a safety zone around the obstacles skirted by the planning algorithm.

3.3 A* Graph Search algorithm

The new obstacle boundaries, as well as the goal position, are then passed to the final function, which actually computes the locally optimal path, using a modified A* search [8] on the local tangent graph. A graph search mechanism is required for this implementation due to the fact that the visibility polygon is not strictly star-shaped, that is, the rover is able to see behind many motion-blocking obstacles. Thus, the resultant LTG is richer than the version in the original TangentBug algorithm. The current rover position and the current Euclidean distance from the goal is placed on the Open list to initialise the search. Then the search progresses as follows: the Open list is sorted by the total path length to the goal, and the entry on the top of the list (with shortest path length) is expanded, and placed on the bottom of the Closed list. Expansion consists of searching for all nodes adjacent to the current node on the LTG. (Note that the LTG is computed over the entire visible region, and since only a single panoramic wedge facing the goal is used, all nodes of the LTG are closer to the goal than the current rover position. Thus, although the locally optimal path may backtrack within the visible polygon--since the visible polygon is no longer strictly a star-shaped set--the ultimate endpoint of the local path is guaranteed to be closer to the goal than the current rover position.) Once these nodes are found, their total path length is computed by adding the "backward" cost--the cumulative lengths of the edges traversed to reach the new node--and the direct distance from the new node to the goal, d(V,G), and they are added to the Open list. The Open list is sorted, and the process begins again. The process halts when the goal point is added to the Open list. The path is then "read out" from the Closed list, skipping noncontiguous points.

4. Experimental Results

The algorithm as implemented for Rocky7 and described above has been tested in several configurations. First, it was tested as a single-step planner--i.e., the start and goal positions were contained in a single panoramic wedge--using imagery collected in the "MarsYard", an outdoor testing facility at JPL designed to mimic the rock distributions seen by the Viking landers on Mars. The functions were implemented in C and run on a Sparc 10 workstation. A typical path generated in this manner is shown in Fig. 5. Next, the algorithm was tested with Rocky7: the planner was used as a single-step, off-line planner which was then executed by the rover: the rover generated the stereo imagery and the range image, which were then used by the off-line C functions to generate a sequence of waypoints which were then used to command Rocky7. These tests revealed unfortunate interaction between the TangentBug planner and Rocky7's current waypoint-based heuristic path executor (the "Go To Waypoint" algorithm described in [11]), indicating that a more appropriate path executor is required.

In a new series of tests about to be completed, the TangentBug-derived planner has been ported on-board Rocky7, which runs VxWorks and ControlShell on a 68060 processor, and the path executor has been replaced (until further work can be done on a more suitable execution algorithm) by commands which drive the rover directly to the specified waypoint, without obstacle-avoidance. This replacement can be justified by the fact that for each step, the stereo data returned by the rover's mast cameras is accurate to 0.8 cm within the 4 meter radius used by the planner. Thus, it is unlikely that within this planning radius the rover would drift enough, or that undetected obstacle boundaries would arise, to make obstacle-avoidance in the path executor necessary. The tests will be conducted with the Rocky7 microrover in the JPL "MarsYard", and will test the full capability of the algorithm by allowing the rover to use multiple panoramic wedges (taken from a single position) for planning if necessary, thus allowing the rover to plan around larger obstacles, and to chain together several steps and visibility polygons to reach a distant goal.

5. Future Work

There is much work which remains to be done which will improve the TangentBug algorithm for use as a planner for long-range (several hundred meter) traverses through planetary terrain. Among issues soon to be addressed are immediate extensions to the planner as implemented on Rocky7 to take advantage of the full capabilities of both the TangentBug algorithm and of the Rocky7 mast cameras, as well as developing a concept of traversability to better be able to utilise the rover's mobility characteristics.

5.1 TangentBug implementation extensions

In the short-term, the current implementation of TangentBug on Rocky7 will be extended to round out the algorithm's capabilities. The boundary-following behaviour will be implemented, for example, as well as the ability to designate goals using means other than strictly coordinate points, e.g. heading and distance. After each step, the convergence proof for the implemented algorithm will be analysed to ensure completeness.

5.2 Traversability

Work has recently been started on incorporating a concept of traversability into the implemented TangentBug planner. As can be seen in the example in Fig. 5, a string of rocks across the far edge of the wedge has blocked off any progress in this direction. Although it may be possible that a clear path can be found by looking either to the left or right, it may also be true that it may be desirable, given the sensed enviroment, to have the rover simply drive over the rocks blocking its path, if possible. It should be pointed out that it is not always desirable to have the rover run roughshod over rocks blocking its path. The current model for obstacles sensed by the stereo vision system (as described above) is a step and slope model: objects exceeding a certain height threshold and a given slope threshold are marked as obstacles. Thus, obstacles are described as those objects which exceed the step-climbing ability of the rover mobility system. Sensor noise and the simple model conspire to make the obstacle sensing algorithm quite conservative. However, given only this knowledge about its environment, the rover must conclude that those areas marked as obstacles are off-limits. Thus there is a need for an analysis of traversability issues and movement away from the simple binary notion of obstacles discussed earlier. Indeed, our firsthand experience with driving the Sojourner rover on Mars has revealed many cases when a binary obstacle model has resulted in halted motion, often leaving the rover in an undesirable configuration.

Two approaches are being explored to extend TangentBug to allow for these more generalised obstacles. Both involve defining the obstacles as a set of weighted regions, rather like a "potential field" approach, where the weights reflect the "costs" associated with traversing the given region. For example, define a function $\alpha(q, q)$ which assigns a cost to each configuration, based upon the

properties of the terrain and mobility characteristics of the robot. The first approach calls the planner repeatedly for each step, where for each iteration the cost threshold associated with what is considered "free space" is raised until either a path is found or the threshold cannot be raised further without increasing the risk to the rover to undesirable levels. The second approach utilises the TangentBug planner once for each step, using as obstacles only those regions marked as "untraversable", and then warps the generated local path according to $\alpha(q,q)$, using calculus of variations techniques, to produce a path which minimises its associated cost.

6. Conclusion

The field of planetary exploration offers a rich environment for mobile robotics research, containing as it does many complicated issues which must be tackled in order to produce successful missions. In particular, this paper addresses some of the issues related to autonomous path-planning for planetary rovers designed to traverse hundreds of meters between uplink opportunities. A sensor-based adaptation of a classical planner with desirable properties has been implemented and tested on the Rocky7 prototype microrover, and several extensions to the current implementation have been proposed, including the vital area of incorporating traversability into planning decisions. With these tools, mobile robots will prove to be an even more useful and robust addition to the techniques available for planetary exploration.

7. Acknowledgements

The work described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

We would like to acknowledge the team members of the Long Range Science Rover task, without whom this work would have been impossible, as well as the Mars Pathfinder Microrover Flight Experiment team, for inspiration and for flight experience with a rover. Sharon would particularly like to thank Todd Litwin, Andrew Mishkin, and Rich Petras for their invaluable assistance with this work.

8. References

[1] Brooks, R. and Flynn, A., "A robust layered control system for a mobile robot," *IEEE Trans. Robotics Automat.*, vol. 2, no. 1, 1986.

[2] Choset, H., Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph. Ph.D. thesis, California Inst. of Tech., 1996.

[3] Kamon, I., Rivlin, E. and Rimon, E., "A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots," in *Proc. IEEE Conf. Robotics Automat.*, 1996.

[4] Kelly, A.J., "RANGER--An Intelligent Predictive Controller for Unmanned Ground Vehicles." The Robotics Institute, Carnegie Mellon, 1994.

[5] Latombe, J.-C., Robot Motion Planning. Kluwer Academic Publishers, 1991.

[6] Matthies, L. and Grandjean, P., "Stochastic Performance Modeling and Evaluation of Obstacle Detectability with Imaging Range Sensors," *IEEE Trans. Robotics Automat.*, vol. 10, no. 6, 1994.

[7] Mishkin, A., Morrison, J., Nguyen, T., Stone, H. and Cooper, B., "Operations and Autonomy of the Mars Pathfinder Microrover," submitted to *IEEE Aerospace Conf.*, 1998.

[8] Rankin, A.L., Path Planning and Path Execution Software for an Autonomous Nonholonomic Robot Vehicle. Master's thesis, University of Florida, 1993.

[9] Stentz, A. and Hebert, M., "A Complete Navigation System for Goal Acquisition in Unknown Environments," *Autonomous Robots*, 2, 1995.

[10] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments," in *Proc. IEEE Conf. Robotics Automat.*, 1994.

[11] Volpe, R., Balaram, J., Ohm, T. and Ivlev, R., "The Rocky 7 Mars Rover Prototype," in *Proc. IEEE/RSJ Conf. Intelligent Robots and Sys.*, 1996.